

04-20-00

PTO/SB/05 (12/97)

Please type a plus sign (+) inside this box → ☐

Approved for use through 9/30/00. OMB 0651-0032

Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

**UTILITY  
PATENT APPLICATION  
TRANSMITTAL**

(Only for new nonprovisional applications under 37 CFR 1.53(b))

|  |               |             |    |
|--|---------------|-------------|----|
| Attorney Docket No.                            | ST9-99-145    | Total Pages | 78 |
| First Named Inventor or Application Identifier |               |             |    |
| Shyh-Mei Ho                                    |               |             |    |
| Express Mail Label No.                         | EL487543635US |             |    |

**APPLICATION ELEMENTS**

See MPEP chapter 600 concerning utility patent application contents.

1. ☒ Fee Transmittal Form  
(Submit an original, and a duplicate for fee processing)
2. ☒ Specification [Total Pages **53**]  
(preferred arrangement set forth below)
  - Descriptive title of the Invention
  - Cross References to Related Application
  - Statement Regarding Fed sponsored R & D
  - Reference to Microfiche Appendix
  - Background of the Invention
  - Brief Summary of the Invention
  - Brief Description of the Drawings (if filed)
  - Detailed Description
  - Claim(s)
  - Abstract of the Disclosure
3. ☒ Drawing(s) (35 USC 113) [Total Sheets **11**]
4. Oath or Declaration [Total Pages **4**]
  - a. ☒ Newly executed (original or copy)
  - b. ☐ Copy from a prior application (37 CFR 1.63(d))  
(for continuation /divisional with Box 17 completed)  
[Note Box 5 below]
    - i. ☐ **DELETION OF INVENTOR(S)**  
Signed statement attached deleting inventor(s)  
named in prior application, see 37 CFR  
1.63(d)(2) and 1.33(b).
5. ☐ Incorporation by Reference (useable if Box 4b is checked)  
The entire disclosure of the prior application, from which  
a copy of the oath or declaration is supplied under Box  
4b is considered as being part of the disclosure of the  
accompanying application and is hereby incorporated by  
reference therein.

ADDRESS TO: Assistant Commissioner for Patents  
Box Patent Application  
Washington, DC 20231

6. ☐ Microfiche Computer Program (Appendix)
7. Nucleotide and/or Amino Acid Sequence Submission (if  
applicable, all necessary)
  - a. ☐ Computer Readable Copy
  - b. ☐ Paper Copy (identical to computer copy)
  - c. ☐ Statement verifying identify of above copies

**ACCOMPANYING APPLICATION PARTS**

8. ☒ Assignment Papers (cover sheet & document(s))
9. ☐ 37 CFR 3.73(b) Statement ☐ Power of Attorney  
(when there is an assignee)
10. ☐ English Translation Document (if applicable)
11. ☒ Information Disclosure ☒ Copies of IDS  
Statement (IDS)/PTO-1449 Citations
12. ☐ Preliminary Amendment
13. ☒ Return Receipt Postcard (MPEP 503)  
(Should be specifically itemized)
14. ☐ Small Entity ☐ Statement filed in prior application,  
Statement(s) Status still proper and desired
15. ☐ Certified Copy of Priority Document(s)  
if foreign priority is claimed
16. ☒ Other: Express Mail Certificate

17. If a **CONTINUING APPLICATION**, check appropriate box and supply the requisite information:☐ Continuation ☐ Divisional ☒ Continuation-in-part (CIP) of prior application No.: 60/151,482**18. CORRESPONDENCE ADDRESS**☒ Customer Number or Bar Code Label

21552

or ☐ Correspondence address below

|         |                     |           |  |     |  |
|---------|---------------------|-----------|--|-----|--|
| NAME    | Kory D. Christensen |           |  |     |  |
| ADDRESS |                     |           |  |     |  |
| CITY    |                     | STATE     |  | ZIP |  |
| COUNTRY |                     | TELEPHONE |  | FAX |  |

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, Patent and Trademark Office, Washington, D.C. 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Washington, D.C. 20231.

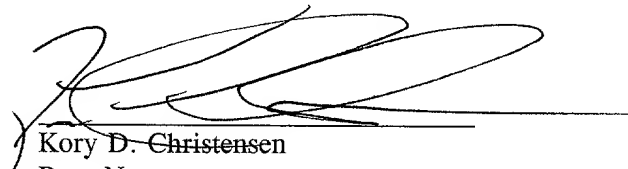
CERTIFICATE OF MAILING BY "EXPRESS MAIL"

"Express Mail" Mailing Label No.: EL487543635US

Date of Deposit: April 19, 2000

I hereby certify that this patent application in the name of Shyh-Mei Ho and Holger Juschkewitz for REPRESENTING IMS TRANSACTION DEFINITIONS AS XML DOCUMENTS, together with the drawings, a joint signature Declaration, Power of Attorney, and Petition, an Assignment, a Recordation Form Cover Sheet, a Fee Transmittal, a Utility Form Cover Sheet, an Information Disclosure Statement, Form PTO-1449, Cited References, and Check No. 13360 for \$748.00 are being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 C.F.R. § 1.10 on the date indicated above in an envelope addressed to Box Patent Application, Assistant Commissioner for Patents, Washington, D.C. 20231.

Respectfully submitted,

  
Kory D. Christensen  
Reg. No.  
Attorney for Applicant

Date: April 19, 2000

MADSON & METCALF  
Gateway Tower West  
15 West South Temple, Suite 900  
Salt Lake City, Utah 84101  
Telephone: 801/537-1700

Express Mailing Label No. EL487543635US

PATENT APPLICATION

Docket No. 3000.2.38

IBM No. ST9-99-145

UNITED STATES PATENT APPLICATION

of

SHYH-MEI HO

and

HOLGER JUSCHKEWITZ

for

REPRESENTING IMS TRANSACTION DEFINITIONS AS XML  
DOCUMENTS

MADSON & METCALF, P.C.

ATTORNEYS AT LAW  
900 GATEWAY TOWER WEST  
15 WEST SOUTH TEMPLE  
SALT LAKE CITY, UTAH 84101

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21

- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21

4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21

- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21

9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21

10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21

14  
15  
16  
17  
18  
19  
20  
21

15  
16  
17  
18  
19  
20  
21

**MADSON & METCALF, P.C.**  
ATTORNEYS AT LAW  
900 GATEWAY TOWER WEST  
15 WEST SOUTH TEMPLE  
SALT LAKE CITY, UTAH 84101

## Relevant Technology

With the explosive growth of the Internet, most of the world's computer systems are now interconnected or capable of being interconnected. However, in order to share data, the systems need to understand each other's data formats. In recent years, the computer industry has evolved at such a rapid pace that systems developed only a few years apart use vastly different and incompatible formats. Such incompatibility problems tend to increase with the "age" differences of the systems.

The Information Management System (IMS) is one of the oldest and perhaps the most popular transaction processing (TP) systems. A TP system supervises the sharing of resources for concurrently processing multiple transactions, such as queries to a database. Anyone who has ever used an ATM, rented a car, or booked a flight, has probably used IMS.

IMS was developed by IBM the 1960's as a inventory tracking system for the U.S. moon landing effort. Today, interfacing IMS with newer systems, particularly with systems of different manufactures over the Internet, is problematic.

As illustrated in Figure 1, an IMS typically includes two major components: an IMS Transaction Monitor (IMS/TM) 12, which is responsible for scheduling, authorization, presentation services and operational functions, and a hierarchical database 14, DL/I. Both components are independently configurable. For example the IMS/TM 12 can use a relational database, such as DB/2, rather than

1 DL/I. The various components of an IMS 10 communicate via the MVS operating  
2 system 16.

3 As illustrated Figure 2, the architecture of IMS is divided into four regions. For  
4 example, a Message Processing Region (MPR) 20 is used to execute message-driven  
5 applications 18. Execution of applications 18 in this region 20 is triggered by  
6 incoming messages, such as those received from a terminal.

7 By contrast, applications 18 in a Batch Message Processing (BMP) 22 region  
8 are not message driven. They are usually scheduled to run at times of low system  
9 activity, such as nights and weekends. Typically, such applications 18 perform a  
10 number of predefined operations, after which they immediately terminate.

11 An Interactive Fast Path (IFP) 24 region allows fast and simple requests to the  
12 database 14. Applications 18 operating in the IFP 24 bypass the normal scheduling  
13 mechanism, providing relatively fast response times. In general, IFP applications  
14 18 stay resident even if they are not needed.

15 An IMS Control Region (IMSCCTL) 26 is responsible for all TP tasks, as well  
16 as for controlling all dependent regions (MPR 20, BMP 22, and IFP 24). Essentially,  
17 the IMS Control Region 26 has three main responsibilities: telecommunications,  
18 message scheduling, and logging/restart.

19 For example, IMSCCTL 26 controls the connected terminals 28 (illustrated in  
20 Figure 3), receiving/sending messages from/to the terminals 28. Moreover,  
21

1 IMSCTL 26 logs all transactions in order to provide the capability of undoing non-  
2 committed transactions in the event of a system failure.

3 In addition, every time IMSCTL 26 receives a message from a terminal 28, it  
4 schedules an application 18 to process the message. IMSCTL 26 identifies the  
5 desired application 18 and puts the message in the application's message queue 30.  
6 The application 18 processes the requests in its message queue 30 and responds to  
7 the originating terminal 28 by placing the response in the terminal's message queue  
8 32.

9 As illustrated in Figure 4, an IMS 10 obtains all of its information about the  
10 structure and behavior of its components (applications, databases, transactions, etc.)  
11 from macro statements 34 (hereinafter "macros"). Certain macros 34 are referred  
12 to as "transaction definitions" 35 because they define how transactions are  
13 processed.

14 For example, as shown in Figure 4, an application (APPLCTN) macro 36  
15 defines the behavior of a particular IMS application 18. An APPLCTN macro 36  
16 exists for each application 18 in the IMS 10, and defines, for example, the  
17 application's name, resource requirements, and appearance.

18 An APPLCTN macro 36 is followed by a zero or more (TRANSACT) macros  
19 38, which define the various transactions applicable to the application 18. A  
20 TRANSACT macro 38 specifies the appearance of a transaction to be performed by  
21 an application 18, identifying whether the transaction is IMS exclusive, IMS Fast

1 Path potential or IMS Fast Path exclusive. Furthermore, a TRANSACT macro 38  
2 specifies the transaction code that causes the application 18 named in the APPLCTN  
3 macro 36 to be scheduled for execution in an IMS processing region.

4 Currently, IMS is only capable of processing transactions previously defined  
5 by the APPLCNT and TRANSACT macros 36, 38. A user may not initiate arbitrary  
6 transactions, such queries of the database 14, that have not been previously defined.  
7 Moreover, in order to change the above-described macros, one must initiate a  
8 process called "system generation," which necessitates shutting down the IMS 10  
9 for a period of time.

10 In addition, the above-described IMS macros 36, 38 have a proprietary  
11 format, which is a detriment in interfacing with remotely located systems from  
12 different vendors. Currently, the dominant Internet format is the HyperText  
13 Markup Language (HTML), a variant of the eXtensible Markup Language (XML).  
14 Providing a technique for delivering IMS transactions definitions to an IMS 10 using  
15 interchangeable documents, such as XML documents, would be a first step in being  
16 able to initiate arbitrary IMS transactions over the Internet.

17 Accordingly, what is needed is a system and method for representing IMS  
18 transaction definitions in an interchangeable format, such as XML. What is also  
19 needed is a system and method for communicating with an IMS 10 using XML  
20 documents. In addition, what is needed is a system and method for creating a  
21



1 Document Type Definition (DTD) for use by a parser in converting between IMS  
2 transaction definitions and XML documents.

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

## SUMMARY OF THE INVENTION

The present invention solves many or all of the foregoing problems by providing a system and method for communicating with an Information Management System (IMS) using eXtensible Markup Language (XML) documents.

In one aspect of the invention, a method includes the steps of receiving a document comprising an IMS transaction definition encoded in XML; obtaining a Document Type Definition (DTD) specifying rules for decoding the IMS transaction definition; parsing the XML document using the DTD to decode the IMS transaction definition; and providing the decoded IMS transaction definition to the IMS.

In another aspect of the invention, a method includes the steps of modeling an IMS transaction definition in a Universal Modeling Language (UML) to produce a UML object model; and processing the UML object model using an XML Metadata Interchange (XMI) utility to create the DTD.

In yet another aspect of the invention, a method includes the steps of obtaining an IMS transaction definition; obtaining a Document Type Definition (DTD) specifying rules for encoding the IMS transaction definition; and parsing the IMS transaction definition with the DTD to encode the IMS transaction definition in an XML document.

In still another aspect, a system includes a document reception module configured to receive a document comprising an IMS transaction definition encoded in XML; a parser configured to obtain a Document Type Definition (DTD) specifying

1 rules for decoding the IMS transaction definition and further configured to parse the  
2 XML document using the DTD to decode the IMS transaction definition; and an IMS  
3 interface module configured to provide the decoded IMS transaction definition to  
4 the IMS.

5 In another aspect of the invention, a system includes a modeling tool  
6 configured to model an IMS transaction definition in a Universal Modeling  
7 Language (UML) to produce a UML object model; and an XML Metadata  
8 Interchange (XMI) utility configured to process the UML object model to generate  
9 the DTD and a number of access classes.

10 These and other objects, features, and advantages of the present invention  
11 will become more fully apparent from the following description and appended  
12 claims, or may be learned by the practice of the invention as set forth hereinafter.

**BRIEF DESCRIPTION OF THE DRAWINGS**

The present invention is more fully disclosed in the following specification, with reference to the accompanying drawings, in which:

Figure 1 is a schematic block diagram of an Information Management System (IMS);

Figure 2 is a schematic block diagram of IMS processing regions;

Figure 3 is a schematic block diagram of message processing within an IMS;

Figure 4 is a schematic block diagram of IMS macros including two IMS transaction definitions;

Figure 5 is a schematic block diagram of a system for communicating with an IMS using eXtensible Markup Language (XML) documents according to an embodiment of the invention;

Figure 6 is a schematic block diagram of a system for generating XML documents from IMS transaction definitions according to an embodiment of the invention;;

Figure 7 is a schematic block diagram of a system for generating a parser, a number of access classes, and a Document Type Definition (DTD) for IMS transaction definitions according to an embodiment of the invention;

Figure 8 is a schematic block diagram of a Universal Modeling Language (UML) object model representing IMS transaction definitions according to an embodiment of the invention;

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21

Figure 9 is a schematic flowchart of a method for communicating with an IMS using XML documents according to an embodiment of the invention;

Figure 10 is a schematic block diagram showing a conversion between an XML document and an IMS transaction definition according to an embodiment of the invention;

Figure 11 is a schematic flowchart of a method for converting an IMS transaction definition into an XML document according to an embodiment of the invention;

Figure 12 is a schematic block diagram showing a conversion between an IMS transaction definition into an XML document according to an embodiment of the invention; and

Figure 13 is a schematic flowchart of a method for generating a parser, a number of access classes, and a DTD for IMS transaction definitions according to an embodiment of the invention.

1           **DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS**

2           Certain presently preferred embodiments of the invention are now described  
3 with reference to the Figures, where like reference numbers indicate identical or  
4 functionally similar elements. The components of the present invention, as  
5 generally described and illustrated in the Figures, may be implemented in a variety  
6 of configurations. Thus, the following more detailed description of the  
7 embodiments of the system and method of the present invention, as represented in  
8 the Figures, is not intended to limit the scope of the invention, as claimed, but is  
9 merely representative of presently preferred embodiments of the invention.

10           Throughout the following description, various system components are  
11 referred to as "modules," "units," "utilities," "tools," or the like. In certain  
12 embodiments, these components may be implemented as software, hardware,  
13 firmware, or any combination thereof.

14           For example, as used herein, a module may include any type of computer  
15 instruction or computer executable code located within a memory device and/or  
16 transmitted as electronic signals over a system bus or network. An identified  
17 module may include, for instance, one or more physical or logical blocks of  
18 computer instructions, which may be embodied within one or more objects,  
19 procedures, functions, or the like.

20           The identified modules need not be located physically together, but may  
21 include disparate instructions stored at different memory locations, which together

1 implement the described logical functionality of the module. Indeed, a module may  
2 include a single instruction, or many instructions, and may even be distributed  
3 among several discrete code segments, within different programs, and across  
4 several memory devices.

5 Figure 5 illustrates a schematic block diagram of a system 40 for  
6 communicating with an Information Management System (IMS) 10 using  
7 interchangeable documents according a presently preferred embodiment of the  
8 invention. The system 40 may include a Web server 46, which may receive a  
9 specially-encoded document 44 from a Web browser 42 via the Internet 48.

10 Various Web servers 46 may be used, such as the Windows NT Server™,  
11 available from Microsoft Corporation. Likewise, the Web browser 42 may be  
12 implemented using a conventional navigation tool for accessing the World Wide  
13 Web (WWW), such as Microsoft Internet Explorer™ or Netscape Navigator™.

14 The document 44 may be encoded to represent one or more IMS transaction  
15 definitions 35, such as APPLCTN or TRANSACT macros 36, 38, using an open  
16 document format, such as the eXtensible Markup Language (XML). Techniques for  
17 encoding IMS transaction definitions 35 in XML are described more fully hereafter.  
18 XML is similar to the HyperText Markup Language (HTML), but provides the  
19 ability to define custom tags by means of Document Type Definitions (DTDs).

20 In various embodiments, the system 40 also includes an IMS gateway 50,  
21 which may provide an interface between the Web server 46 and the IMS 10. The

1 IMS gateway 50 may include an XML to Macro parser 52, which may parse the XML  
2 document 44 using a specialized DTD 54 to decode the IMS transaction definition  
3 35. A system and method for generating the DTD 54 and the parser 52 is described  
4 more fully hereafter. After the transaction definition 35 is decoded, the IMS  
5 gateway 50 may provide the same to the IMS 10.

6 In alternative embodiments, the system 40 may be used in reverse to generate  
7 an XML document 44 from an IMS transaction definition 35, as illustrated Figure 6.  
8 For example, a transaction definition 35 may be obtained from the IMS 10 or another  
9 source. Thereafter, a Macro to XML parser 53 within the IMS gateway 50 may use  
10 the DTD 54 to encode the transaction definition 35 as an XML document 44. In  
11 certain embodiments, two different parsers 52, 53 are used, one for encoding and  
12 one for decoding a transaction definition 35, although the invention is not limited  
13 in this respect. After the XML document 44 is generated, the Web server 46 may  
14 transmit the same to a Web browser 42 or the like via the Internet 48.

15 Figure 7 illustrates a system 60 for generating the DTD 54 and a number of  
16 access classes 76 and for creating the parsers 52, 53 based on the DTD 54 and the  
17 access classes 76. In various embodiments, the system 60 includes a Uniform  
18 Modeling Language (UML) modeling tool 62. The UML modeling tool 62 may be  
19 used to produce a UML object model 64 to represent the transaction definitions 35.  
20 In one embodiment, the UML modeling tool 62 is Rational Rose™, a visual modeling  
21 tool available from Rational Software. Appendix A includes a UML Model Report



1 generated by Rational Rose, as well as a description of various UML classes  
2 according to an embodiment of the invention.

3 UML is a language for specifying, visualizing, constructing, and  
4 documenting software systems, as well as business models and the like. UML is  
5 capable of representing the static, dynamic, environmental, and organizational  
6 aspects of almost any system.

7 Figure 8 illustrates a UML object model 64 of the APPLCTN and TRANSACT  
8 macros 36, 38, as displayed by the UML modeling tool 62. In one embodiment, the  
9 model 64 includes an *ApplicationControlMacro* object 66 to represent the APPLCTN  
10 macro 36. Each parameter of the APPLCTN macro 36 may be represented by an  
11 attribute 67 of the *ApplicationControlMacro* object 66. For example, the PSBName  
12 attribute represents a parameter in the APPLCTN macro 36 for the Program  
13 Specification Block (PSB) name.

14 As illustrated, the *ApplicationControlMacro* object 66 may be linked to zero or  
15 more *TransactionControlMacro* objects 68. In one embodiment, a  
16 *TransactionControlMacro* object 68 represents the TRANSACT macro 38, each  
17 attribute 69 of the object 68 corresponding to a parameter of the macro 38.

18 In addition, the UML object model 64 may include a number of enumeration  
19 stereotypes 70, which are type definitions of attributes 67, 69 having a fixed set of  
20 values. For example, an attribute 67 of the type, *TscheduleType*, may only have a  
21 value of "Serial" or "Parallel" in certain embodiments. Likewise, the UML object

1 model 64 may include a number of primitive stereotypes 72, which represent the  
2 basic variable types of the model 64, such as strings, unsigned integers, and the like.

3 Referring again to Figure 7, the system 60 may also include an XML Metadata  
4 Interchange (XMI) utility 74, such as XMI Toolkit™, available from IBM. XMI is an  
5 open standard released by the Object Management Group (OMG) for simplifying  
6 the exchange of data and metadata between different products from different  
7 vendors. IBM's XMI Toolkit is written entirely in Java and offers interfaces to  
8 facilitate incorporation into other projects or products.

9 In various embodiments, the XMI utility 74 automatically generates the DTD  
10 54 from the UML object model 64. An example of a DTD 54 generated by the XMI  
11 utility 74 according to an embodiment of the invention is provided in Appendix B.

12 In addition, the XMI utility 74 may create a plurality of Java XMI document access  
13 classes 76, which are used by the parsers 52, 53 to parse and transform the document  
14 44 based on the generated DTD 54. The access classes 76 provide an application  
15 programming interface (API) to allow the parsers 52, 53 to export a transaction  
16 definition 35 to an XML document 44 and to import an XML document 44 to a  
17 transaction definition 35.

18 The system 60 may also include a parser generator 78, such as the Java  
19 Compiler Compiler (JavaCC) 1.1, available from Sun Microsystems, although a  
20 variety of parser generators 78 may be used. JavaCC converts a grammar  
21 specification into a Java program that can recognize matches to the grammar. In

1 certain embodiments, the parser generator 78 is only used in the creation of the  
2 parser 53 (Macro to XML), since the XMI access classes 76 generated by the XMI  
3 utility 74 may be used to directly import the entire content of the document 44 by  
4 invoking a single method. However, other techniques for parser generation may  
5 be used in alternative embodiments.

6 Referring now to Figure 9, a schematic flowchart illustrates a method 80 for  
7 communicating with an IMS 10 using XML documents 44. The method may begin  
8 by receiving 82 a document 44 at an IMS gateway 50. The document 44 may be  
9 received, for example, from a Web server 46 attached to the Internet 48.

10 After the document 44 is received 82, the method 80 may continue by  
11 obtaining 84 a DTD 54 for IMS transaction definitions 35, such as the DTD 54  
12 illustrated in Appendix B. Thereafter, the method may continue by parsing 86 the  
13 document 44 using the DTD 54 to decode the IMS transaction definition 35, and  
14 providing 88 the same to the IMS 10.

15 Figure 10 illustrates the above-described process. For example, a portion of  
16 an XML-encoded document 44 representing an APPLCTN macro 36 is shown. A  
17 parser 52 obtains the DTD 54 and parses the document 44 to decode the macro 36,  
18 i.e. "APPLCTN PSB=ABCD,PGMTYPE=(TP,1)."

19 Figure 11 illustrates a method 90 for generating an XML document 44 from  
20 an IMS transaction definition 35 according to an embodiment of the invention. The  
21

1 method 90 may begin by obtaining 92 an IMS transaction definition 35 from an IMS  
2 10 or another source.

3 After the transaction definition 35 is obtained 92, the method 90 may  
4 continue by obtaining 94 a DTD 54 for IMS transaction definitions 35, as illustrated  
5 in Appendix B. Thereafter, the method 90 may continue by parsing 96 and  
6 transforming the IMS transaction definition 35 using the DTD 54 to encode the same  
7 as an XML document 44. Finally, the XML document 44 may be sent 98 to an XMI-  
8 enabled device, such as a Web browser 42 or the like.

9 Figure 12 illustrates the above-described process. A subset of an APPLCTN  
10 macro 36 is shown, i.e. "APPLCTN PSB=ABCD,FPATH=30720." The parser 53  
11 obtains the DTD 54 and parses and transforms the macro 36 to encode the same as  
12 an XML document 44, as illustrated.

13 Figure 13 illustrates a method 100 for generating a DTD 54 and a parser 53  
14 in accordance with an embodiment of the invention. The method 100 may begin by  
15 modeling 102 an IMS transaction definition 35 in a Universal Modeling Language  
16 (UML) to produce a UML object model 64. As previously noted, a visual modeling  
17 tool, such as Rational Rose, may be used for this purpose.

18 After the transaction definition 35 is modeled 102, the method 100 may  
19 continue by processing 104 the UML object model 64 using an XMI utility 74, such  
20 as IBM's XMI Toolkit, to generate the DTD 54 and various document access classes  
21 76.

1 In certain presently preferred embodiments, the method 100 may continue  
2 by processing the DTD 54 using a parser generator 78, such as the Java Compiler  
3 Compiler (JavaCC), in order to generate the parser 53.

4 Based on the foregoing, the present invention offers a number of advantages  
5 over conventional IMS interface systems. For example, the present invention  
6 communicates with an IMS 10 using openly interchangeable documents, such as  
7 XML documents 44. The documents 44 may comply with the latest XMI standard,  
8 which enables an IMS 10 to exchange data with a variety of XMI-enabled devices,  
9 such as Web browsers 42 and the like. Accordingly, the present invention  
10 overcomes a first hurdle in the goal of facilitating arbitrary IMS transactions via the  
11 Internet.

12 Importantly, XML documents 44 may be easily converted for display on a  
13 variety of computing platform using the emerging XML Style Language (XSL)  
14 standard. As such, the XMI to IMS interface is capable of replacing all other  
15 interfaces between IMS and products from other vendors.

16 The present invention may be embodied in other specific forms without  
17 departing from its scope or essential characteristics. The described embodiments  
18 are to be considered in all respects only as illustrative and not restrictive. The scope  
19 of the invention is, therefore, indicated by the appended claims rather than by the  
20 foregoing description. All changes which come within the meaning and range of  
21 equivalency of the claims are to be embraced within their scope.

**ATTORNEYS AT LAW  
900 GATEWAY TOWER WEST  
15 WEST SOUTH TEMPLE  
SALT LAKE CITY, UTAH 84101**

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21

What is claimed is:

## UML Model Report and Class Descriptions

The following is a UML model report generated by Rational Rose, as well as descriptions of various UML classes according to one embodiment of the invention.

### *SubComponents*

None

## ApplicationControlMacro

TransactionControlMacro

TResidency

TLanguage

## TScheduleType

TProgramType

## Boolean

TMode

## MessageType

String

TSysID

TFPBuffer

TClass

TPriority

## TScheduleOption

## TParallelLimit

TMaxRegions

## ProcessLimitType

TResponse

### *ApplicationControlMacro Class*

The `ApplicationControlMacro` defines the program resource requirements for application programs that run under the control of the IMS DB/DC environment, as well as for applications that access databases through DBCTL. When combined with one or more `TransactionControlMacros`, the `ApplicationControlMacro` defines the scheduling and resource requirements for an application program. On the `ApplicationControlMacro`, one only describes programs that operate in message processing regions, Fast Path message-driven program regions, batch message processing regions, or CCTL threads. The `ApplicationControlMacro` is also used to describe application programs that operate in batch processing regions.

## None

**ResidencyOption: TResidency**

The positional parameter `Resident` specifies that the PSB associated with the application program is to be made resident during system initialization. `Resident` and `DynamicOption` (dynamic PSB option) are mutually exclusive. `DynamicOption` and `ScheduleType=Parallel` are also mutually exclusive.

Neither `Resident` nor `DynamicOption` is the default parameter. Rather, if neither `Resident` nor `DynamicOption` is selected on the `ApplicationControlMacro` statement, IMS system initialization causes a BLDL to be performed on the PSB associated with the application being defined. The PSB is not made resident (that is, loaded from the ACBLIB) until the application is scheduled.

**isFPath:** Boolean = False

Specifies whether the application is a Fast Path exclusive application program. If True, the ProgramType parameter is invalid and TransactionControlMacro.isWaitForInput=True is forced.

**FPathBufferSize: TFPBuffer**

Determines the EMH buffer size required to run the transaction and overrides the EML execution parameter.



**GeneratedPSBName: String**

Defining this attribute causes the scheduling process of all environments to generate a PSB containing an I/O PCB and an alternate modifiable PCB. With this attribute coded, one does not need to perform the PSBGEN and the ACBGEN, thus eliminating I/O to the ACBLIB.

GeneratedPSBName generates an I/O PCB named IOPCBbbb; The modifiable, alternate PCB is named TPPCB1bb. With an alternate modifiable PCB, an application can use the CHNG call to change the output destination and send output to a destination other than the input destination.

The GeneratedPSBName and the PSBName attributes are mutually exclusive.

**GeneratedPSBLanguage: TLanguage = Assembler**

Defines the language interface of the application program.

This attribute may only be valid if the GeneratedPSBName is specified.

**ProgramType: TProgramType = TeleProcessing**

Describes the type of application program being defined. The default, TeleProcessing, specifies that IMS schedules the program when messages processed by the program exist in the system. A program defined as Batch can use DL/I in the IMS control program system region and can refer to the message queues. If Batch is coded, all TransactionControlMacros statements that follow are assigned normal and limit priority values of zero.

**ProgramClass: TClass = 1**

This attribute specifies the class to which the transaction codes specified in the TransactionControlMacro statements are to be assigned.

The value should not exceed the value given (by specification or default) on the MAXCLAS keyword of the IMSCTRL macro.

If the transaction code class is defined in the individual TransactionControlMacro statement, this parameter is ignored.

The numeric class parameter should not be specified if ApplicationControlMacro.isFPath=True is specified.

**PSBName: String**

Specifies the name of the PSB associated with this application program definition. Each local PSB name must be unique. A remote and a local application can each be defined as having the same PSB name. This may be required in order to dynamically reassign a transaction from remote to local processing.

The GeneratedPSBName and the PSBName attributes are mutually exclusive. The PSBName is required, if no GeneratedPSBName is specified.

**ScheduleType: TScheduleType = Serial**

- 1 Specifies the application program can be scheduled into more than one message region or batch  
message region simultaneously..
- 2 ScheduleType and ResidencyOption=DynamicOption are mutually exclusive.
- 3 **SysIDRemote: TSysID**
- 4 Specifies in the multiple IMS system configuration, the system identification (SysID) of the  
remote system (that system on which the application executes).
- 5 If the ApplicationControlMacro.SysID parameters are defined, all other parameters of the  
ApplicationControlMacro except the PSB are ignored. If the TransactionControlMacro.SysID  
6 parameters are defined, the ApplicationControlMacro.SysID parameters are ignored. This  
parameter is invalid, if ApplicationControlMacro.isFPath=True is coded.
- 7 If this attribute is specified the SysIDLocal attribute must be specified, too.
- 8 **SysIDLocal: TSysID**
- 9 Specifies in the multiple IMS system configuration, the system identification (SysID) of the local  
system (the originating system to which the response are returned).
- 10 If the ApplicationControlMacro.SysID parameters are defined, all other parameters of the  
ApplicationControlMacro except the ApplicationControlMacro.PSB are ignored. If the  
TransactionControlMacro.SysID parameters are defined, the ApplicationControlMacro.SysID  
11 parameters asr ignored. This attribute is invalid, if ApplicationControlMacro.isFPath=True is  
coded.
- 12 If this attribute is specified the SysID\_Remote attribute must be specified, too.
- 13 **Inherited Attributes**
- 14 **Associations**
- 15 **TRANSACT with TransactionControlMacro**
- 16 ***TransactionControlMacro Class***
- 17 **Introduction**
- 18 The TRANSACT macro statement is used one or more times with each ApplicationControlMacro  
statement to identify a transaction as IMS exclusive, IMS Fast Path potential or IMS Fast Path  
exclusive. It specifies the transaction codes that cause the application program named in the  
preceding ApplicationControlMacro to be scheduled for execution in an IMS message processing  
region. It also provides the IMS control program with information that influences the application  
20 program scheduling algorithm. It can define a message editing routine.
- 21 **Base Classes**

1 None

2 **Attributes**

3 **TransactionCode:** String

4 Specifies the transaction code associated with this TransactionControlMacro statement.

5 **isDCLogWriteAhead:** Boolean

6 Specifies whether IMS should perform log write-ahead for recoverable, nonresponse mode input messages and transactions output messages. If not specified in the TransactionControlMacro, the default is the DCLogWriteAhead parameter in the IMSCTRL macro. The DCLogWriteAhead parameter in the TransactionControlMacro overrides the IMSCTRL macro parameter for this transaction.

7 **toUpperCase:** Boolean = True

8 Specifies whether (True) or not (False) the input data is to be translated to uppercase before being presented to the processing program.

9 Specifying True for VTAM terminals prevents the transmission of embedded device control characters.

10 Specifying the attribute requires specifying the EditRoutine attribute.

11 **EditRoutine:** String

12 This attribute allows a user to a transaction input edit routine that edits messages prior to the program receiving the message. The specified edit routine (load module) must reside on the USERLIB data set prior to IMS system definition stage 2 execution. This routine cannot be the same as the one that is used on a LINEGRP or TYPE EDIT parameter.

14 This attribute is invalid on a Fast Path exclusive transaction.

15 **isFPath:** Boolean = False

16 Specifies whether the transaction code is a potential candidate for Fast Path processing. Specifying True is effective only when specified on a TransactionControlMacro that follows an ApplicationControlMacro statement that does not specify isFPath=True; otherwise, the operand is ignored.

18 Fast Path potential transactions must be processed by a user edit/routing exit to determine whether the transaction is actually to be processed by IMS Fast Path. If it is to be processed by IMS Fast Path, the edit/routing exit associates the transaction with a routing code. This routing code identifies which Fast Path application program is to process the transaction.

20 **FPathBufferSize:** TFPBuffer

21 Determines the EMH buffer size required to run the transaction and overrides the EML execution parameter.

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

Specifies whether or not the communication line from the transaction was entered is to be held until a response is received.

The meaning of the values in detail:

**NonResponse:** Specifies that, for terminals specifying or defaulting to `OPTIONS=TRANRESP`, input should not stop after this transaction is entered.

**Response:** Specifies that, for terminals specifying or defaulting to `OPTIONS=TRANRESP`, no further messages are to be allowed after this transaction is entered until this transaction sends a response message back to the terminal. Response mode can be forced or negated by individual terminal definition.

**MessageClass:** `TCLass = 1`

Specifies the class to which this transaction code is to be assigned.

**ParallelLimit:** `TParallelLimit`

Specifies the threshold value to be used when `ApplicationControlMacro.ScheduleType=Parallel` was specified in the preceding `ApplicationControlMacro` instruction. An additional region is scheduled whenever the current transaction enqueue count exceeds the `ParallelLimit` value multiplied by the number of regions currently scheduled for this transaction. If `ParallelLimit` is not specified, the default value of none is assumed, and IMS allows the transaction to be scheduled in only one region at a time.

`ParallelLimit=0` indicates that any input message can cause a new region to be scheduled because the scheduling condition will always be met.

If `ParallelLimit=0` is specified, the `MaxRegions` value should be specified to limit the number of regions that can be scheduled to process a particular transaction.

This operand is not appropriate if a transaction destined for processing in another (remote) system is being specified.

**ProcessLimitCount:** `UnsignedInteger = 65535`

Specifies the number of messages (count) of this transaction code a program can process in a single scheduling.

This field specifies the number of messages sent to the application program by the IMS control program for processing without reloading the application program. If 0 is coded, the maximum number of messages sent to the application is one and the application program is reloaded before receiving a subsequent message. The value 65535 means no limit is being set.

This operand is not appropriate if a transaction destined for processing in another (remote) system is being specified.

**ProcessLimitSeconds:** `UnsignedInteger = 65535`

1 Specifies the amount of time (in seconds) allowable of this transaction code to process a single transaction (or message).

2 If Fast Path is used, the keyword parameter specifies, for a given transaction code, the amount  
3 of time (in hundredths of seconds) the program is allowed to process a single transaction  
4 message. The time represents real time elapses during transaction processing (not accumulated  
task time). Real time is used because the input terminal is in response mode and cannot enter  
another transaction until the response is sent. In this case, the ProcessLimitCount is ignored.

5 The value 0 is invalid.

6 This operand is not appropriate if a transaction destined for processing in another (remote)  
system is being specified.

7 **PriorityNormal: TPriority = 1**

8 Specifies the priority of this transaction when the number of input transactions enqueued and  
waiting to be processed is less than the PriorityLimitCount value.

9 **PriorityLimit: TPriority = 1**

10 Specifies the priority of this transaction to which it is raised when the number of input  
transactions, enqueued and waiting to be processed is equal to or greater then the  
PriorityLimitCount value.

11 **PriorityLimitCount: UnsignedInteger = 65535**

12 Specifies the number that, when compared to the number of input transactions enqueued and  
waiting to be processed, determines whether the normal or limit priority value is assigned to this  
transaction.

13 The value 0 is invalid.

14 **isRouting: Boolean = False**

15 If MSC directed routing is used in a multiple IMS system configuration, specifies whether the  
application program processing a transaction is informed of the system which originated the  
transaction.

16 **ScheduleOption: TScheduleOption = 1**

17 Specifies the scheduling option used for other transactions when this transaction cannot be  
scheduled for internal reasons (database intent or no more space in PSB pool or DMB pool to  
bring in needed blocks).

18 The meaning of the values in detail:

19 1: Schedule only transactions of equal or higher priority in the selected class.

20 2: Schedule higher priority transactions in the selected class.

21 3: Schedule any transaction in the selected class.

4: Skip to the next class and attempt to schedule the highest priority transaction in that class.

This operand is not appropriate if a transaction destined for processing in another (remote) system is being specified.

**SegmentNumber: UnsignedInteger = 0**

Specifies the maximum number of application program output segments that will be allowed into the message queues per Get Unique (GU) call from the application program.

This operand is not appropriate if a transaction destined for processing in another (remote) system is being specified.

**SegmentSize: UnsignedInteger = 0**

Specifies the maximum number of bytes allowed in any one output segment.

This operand is not appropriate if a transaction destined for processing in another (remote) system is being specified.

**isSerial: Boolean = False**

Specifies whether the processing of messages for a given transaction is forced to be serial.

If True is specified, ParallelLimit and MaxRegions must equal zero (they can be omitted). If they are not, False is assumed.

This operand is not appropriate if a transaction destined for processing in another (remote) system is being specified.

**SPASize: TSpaSize**

Defines by inclusion that this transaction is a conversational transaction.

The value specifies the size of the conversational scratchpad area (SPA).

**SPAIsTruncated: Boolean = True**

Specifies whether the truncated data option is used with the scratchpad area. .

**SysIDRemote: TSysID**

Specifies in the multiple IMS system configuration, the system identification (SysID) of the remote system (that system on which the application executes).

If the ApplicationControlMacro.SysID parameters are specified, the TransactionControlMacro.SysID parameters don't need to be specified. If both SysID parameters are specified, the ApplicationControlMacro.SysID parameters are ignored.

**SysIDLocal: TSysID**

Specifies in the multiple IMS system configuration, the system identification (SysID) of the local system (the originating system to which the response are returned).

If the ApplicationControlMacro.SysID parameters are specified, the TransactionControlMacro.SysID parameters don't need to be specified. If both SysID parameters are specified, the ApplicationControlMacro.SysID parameters are ignored.

**isWaitForInput:** Boolean

Specifies whether (True) or not (False) this is a wait-for-input transaction. A message processing or batch processing application program that processes WaitForInput transactions is scheduled and invoked normally. If the transaction to be processed is defined as WaitForInput, the program is allowed to remain in main storage after it has processed the available input messages. The QC status code (no more messages) is returned to the program if the ProcessLimitCount value is reached.

Mode=SingleMessage will be forced, if isWaitForInput=True is coded.

This operand is not appropriate if a transaction destined for processing in another (remote) system is being specified.

#### Inherited Attributes

#### Associations

APPLCTN with ApplicationControlMacro

#### *TResidency enumeration*

##### Attributes

Resident

DynamicOption

#### *TLanguage enumeration*

##### Attributes

Assembler

COBOL

PL/1

PL/I

Pascal

#### *TScheduleType enumeration*

##### Attributes

Serial

Parallel

#### *TProgramType enumeration*

##### Attributes



|    |  |
|----|--|
| 1  | TeleProcessing                                       |
| 2  | Batch  |
| 3  | <i>Boolean enumeration</i>                           |
| 4  | Attributes   |
| 5  | False  |
| 6  | True   |
| 7  | <i>TMode enumeration</i>                             |
| 8  | Attributes   |
| 9  | MultipleMessage                                      |
| 10 | SingleMessage  |
| 11 | <i>MessageType enumeration</i>                       |
| 12 | Attributes   |
| 13 | MULTSEG, NONRESPONSE                                 |
| 14 | SNGLSEG, RESPONSE                                    |
| 15 | <i>String primitive</i>                              |
| 16 | length: 1..8   |
| 17 | valid characters: A through Z, #, \$, @, 0 through 9 |
| 18 | <i>TSysID primitive</i>                              |
| 19 | Valid values: 1..2036                                |
| 20 | <i>TFPBuffer primitive</i>                           |
| 21 | Valid values: 12..30720                              |
| 22 | <i>TClass primitive</i>                              |
| 23 | valid values: 1..255                                 |
| 24 | <i>TPriority primitive</i>                           |
| 25 | valid values: 0..14                                  |
| 26 | <i>TScheduleOption primitive</i>                     |
| 27 | valid values: 1..4                                   |
| 28 | <i>TParallelLimit primitive</i>                      |
| 29 | valid values: 0..32767                               |

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21

***TMaxRegions primitive***

valid values: 0..255

***ProcessLimitType primitive***

valid values: 1..65535

***UnsignedInteger primitive***

valid values: 0..65535

***TSpaSize primitive***

valid values: 16..32767

***TSegment enumeration***

Attributes

MultipleSegment

SingleSegment

***TResponse enumeration***

Attributes

Response

NonResponse

## APPENDIX B

## Document Type Definition (DTD)

The following is a DTD for IMS transaction definitions according to an embodiment of the invention.

```

<?xml version="1.0" encoding="UTF-8" ?>

<!-- XMI Automatic DTD Generation      -->
<!-- Metamodel: macro                  -->
<!-- _____                        -->
<!-- _____                        -->
<!-- XMI is the top-level XML element for XMI transfer text      -->
<!-- _____                        -->

<!ELEMENT XMI (XMI.header, XMI.content?, XMI.difference*,
               XMI.extensions*) >
<ATTLIST XMI
    xmi.version CDATA #FIXED "1.0"
    timestamp CDATA #IMPLIED
    verified (true | false) #IMPLIED
>

<!-- _____                        -->
<!-- _____                        -->
<!-- XMI.header contains documentation and identifies the model, -->
<!-- metamodel, and metamodel          -->
<!-- _____                        -->

<!ELEMENT XMI.header (XMI.documentation?, XMI.model*, XMI.metamodel*,
                      XMI.metamodelmodel*) >

<!-- _____                        -->
<!-- _____                        -->
<!-- documentation for transfer data   -->
<!-- _____                        -->

<!ELEMENT XMI.documentation (#PCDATA | XMI.owner | XMI.contact |
                             XMI.longDescription | XMI.shortDescription |
                             XMI.exporter | XMI.exporterVersion |
                             XMI.notice)* >

<!ELEMENT XMI.owner ANY >

```

```

1  <!--ELEMENT XMI.contact ANY >
2  <!--ELEMENT XMI.longDescription ANY >
3  <!--ELEMENT XMI.shortDescription ANY >
4  <!--ELEMENT XMI.exporter ANY >
5  <!--ELEMENT XMI.exporterVersion ANY >
6  <!--ELEMENT XMI.exporterID ANY >
7  <!--ELEMENT XMI.notice ANY >
8  <!-- _____ -->
9  <!-- _____ -->
10 <!-- XMI.element.att defines the attributes that each XML element
11 <!-- that corresponds to a metamodel class must have to conform to
12 <!-- the XMI specification.
13 <!-- _____ -->
14 <!-- _____ -->
15 <!-- _____ -->
16 <!-- _____ -->
17 <!-- _____ -->
18 <!-- XMI.link.att defines the attributes that each XML element that
19 <!-- corresponds to a metamodel class must have to enable it to
20 <!-- function as a simple XLink as well as refer to model
21 <!-- constructs within the same XMI file.
22 <!-- _____ -->
23 <!-- _____ -->
24 <!-- _____ -->
25 <!-- XMI.model identifies the model(s) being transferred
26 <!-- _____ -->
27 <!-- _____ -->
28 <!-- _____ -->
29 <!-- _____ -->
30 <!-- _____ -->
31 <!-- _____ -->
32 <!-- _____ -->
33 <!-- _____ -->
34 <!-- _____ -->
35 <!-- _____ -->
36 <!-- _____ -->
37 <!-- _____ -->
38 <!-- _____ -->
39 <!-- _____ -->
40 <!-- _____ -->
41 <!-- _____ -->
42 <!-- _____ -->
43 <!-- _____ -->
44 <!-- _____ -->
45 <!-- _____ -->
46 <!-- _____ -->
47 <!-- _____ -->
48 <!-- _____ -->
49 <!-- _____ -->
50 <!-- _____ -->
51 <!-- _____ -->
52 <!-- _____ -->
53 <!-- _____ -->
54 <!-- _____ -->
55 <!-- _____ -->
56 <!-- _____ -->
57 <!-- _____ -->
58 <!-- _____ -->
59 <!-- _____ -->
60 <!-- _____ -->
61 <!-- _____ -->
62 <!-- _____ -->
63 <!-- _____ -->
64 <!-- _____ -->
65 <!-- _____ -->
66 <!-- _____ -->
67 <!-- _____ -->
68 <!-- _____ -->
69 <!-- _____ -->
70 <!-- _____ -->
71 <!-- _____ -->
72 <!-- _____ -->
73 <!-- _____ -->
74 <!-- _____ -->
75 <!-- _____ -->
76 <!-- _____ -->
77 <!-- _____ -->
78 <!-- _____ -->
79 <!-- _____ -->
80 <!-- _____ -->
81 <!-- _____ -->
82 <!-- _____ -->
83 <!-- _____ -->
84 <!-- _____ -->
85 <!-- _____ -->
86 <!-- _____ -->
87 <!-- _____ -->
88 <!-- _____ -->
89 <!-- _____ -->
90 <!-- _____ -->
91 <!-- _____ -->
92 <!-- _____ -->
93 <!-- _____ -->
94 <!-- _____ -->
95 <!-- _____ -->
96 <!-- _____ -->
97 <!-- _____ -->
98 <!-- _____ -->
99 <!-- _____ -->
100 <!-- _____ -->

```

```

1  <!ATTLIST XMI.model
      %XMI.link.att;
      xmi.name  CDATA #REQUIRED
2      xmi.version CDATA #IMPLIED
3  >
4  <!-- _____ -->
5  <!-- XMI.metamodel identifies the metamodel(s) for the transferred
6  <!-- data -->
7  <!-- _____ -->
8  <!ELEMENT XMI.metamodel ANY >
9  <!ATTLIST XMI.metamodel
      %XMI.link.att;
      xmi.name  CDATA #REQUIRED
10     xmi.version CDATA #IMPLIED
11 >
12 <!-- _____ -->
13 <!-- XMI.metametamodel identifies the metametamodel(s) for the
14 <!-- transferred data -->
15 <!-- _____ -->
16 <!ELEMENT XMI.metametamodel ANY >
17 <!ATTLIST XMI.metametamodel
      %XMI.link.att;
      xmi.name  CDATA #REQUIRED
18     xmi.version CDATA #IMPLIED
19 >
20 <!-- _____ -->
21 <!-- XMI.content is the actual data being transferred
22 <!-- _____ -->
23 <!ELEMENT XMI.content ANY >
24 <!-- _____ -->
25 <!-- XMI.extensions contains data to transfer that does not conform
26 <!-- to the metamodel(s) in the header -->
27 <!-- _____ -->
28 <!ELEMENT XMI.extensions ANY >

```

```

1  <!ATTLIST XMI.extensions
    xmi.extender CDATA #REQUIRED
2  >
3  <!-- _____ -->
4  <!-- extension contains information related to a specific model -->
5  <!-- construct that is not defined in the metamodel(s) in the header -->
6  <!-- _____ -->
7  <!ELEMENT XMI.extension ANY >
8  <!ATTLIST XMI.extension
    %XMI.element.att;
    %XMI.link.att;
    xmi.extender CDATA #REQUIRED
    xmi.extenderID CDATA #REQUIRED
9  >
10 <!-- _____ -->
11 <!-- XMI.difference holds XML elements representing differences to a -->
12 <!-- base model -->
13 <!-- _____ -->
14 <!ELEMENT XMI.difference (XMI.difference | XMI.delete | XMI.add |
    XMI.replace)* >
15 <!ATTLIST XMI.difference
    %XMI.element.att;
    %XMI.link.att;
16 >
17 <!-- _____ -->
18 <!-- XMI.delete represents a deletion from a base model -->
19 <!-- _____ -->
20 <!ELEMENT XMI.delete EMPTY >
21 <!ATTLIST XMI.delete
    %XMI.element.att;
    %XMI.link.att;
    >
    <!-- _____ -->
    <!-- _____ -->
    <!-- XMI.add represents an addition to a base model -->

```

```

1  <!-- _____ -->
2  <!ELEMENT XMI.add ANY >
3  <!ATTLIST XMI.add
4      %XMI.element.att;
5      %XMI.link.att;
6      xmi.position CDATA "-1"
7  >
8  <!-- _____ -->
9  <!-- _____ -->
10 <!-- XMI.replace represents the replacement of a model construct
11 <!-- with another model construct in a base model
12 <!-- _____ -->
13 <!-- _____ -->
14 <!ELEMENT XMI.replace ANY >
15 <!ATTLIST XMI.replace
16     %XMI.element.att;
17     %XMI.link.att;
18     xmi.position CDATA "-1"
19 >
20 <!-- _____ -->
21 <!-- _____ -->
22 <!-- XMI.reference may be used to refer to data types not defined in
23 <!-- the metamodel
24 <!-- _____ -->
25 <!-- _____ -->
26 <!ELEMENT XMI.reference ANY >
27 <!ATTLIST XMI.reference
28     %XMI.link.att;
29 >
30 <!-- _____ -->
31 <!-- _____ -->
32 <!-- This section contains the declaration of XML elements
33 <!-- representing data types
34 <!-- _____ -->
35 <!-- _____ -->
36 <!ELEMENT XMI.TypeDefinitions ANY >
37 <!ELEMENT XMI.field ANY >
38 <!ELEMENT XMI.seqItem ANY >
39 <!ELEMENT XMI.octetStream (#PCDATA) >

```

```

1  <!ELEMENT XMI.unionDiscrim ANY >
2  <!ELEMENT XMI.enum EMPTY >
3  <!ATTLIST XMI.enum
4      xmi.value CDATA #REQUIRED
5  >
6  <!ELEMENT XMI.any ANY >
7  <!ATTLIST XMI.any
8      %XMI.link.att;
9      xmi.type CDATA #IMPLIED
10     xmi.name CDATA #IMPLIED
11 >
12 <!ELEMENT XMI.CorbaTypeCode (XMI.CorbaTcAlias | XMI.CorbaTcStruct |
13     XMI.CorbaTcSequence | XMI.CorbaTcArray |
14     XMI.CorbaTcEnum | XMI.CorbaTcUnion |
15     XMI.CorbaTcExcept | XMI.CorbaTcString |
16     XMI.CorbaTcWstring | XMI.CorbaTcShort |
17     XMI.CorbaTcLong | XMI.CorbaTcUshort |
18     XMI.CorbaTcUlong | XMI.CorbaTcFloat |
19     XMI.CorbaTcDouble | XMI.CorbaTcBoolean |
20     XMI.CorbaTcChar | XMI.CorbaTcWchar |
21     XMI.CorbaTcOctet | XMI.CorbaTcAny |
22     XMI.CorbaTcTypeCode | XMI.CorbaTcPrincipal |
23     XMI.CorbaTcNull | XMI.CorbaTcVoid |
24     XMI.CorbaTcLongLong |
25     XMI.CorbaTcLongDouble) >
26 <!ATTLIST XMI.CorbaTypeCode
27     %XMI.element.att;
28 >
29 <!ELEMENT XMI.CorbaTcAlias (XMI.CorbaTypeCode) >
30 <!ATTLIST XMI.CorbaTcAlias
31     xmi.tcName CDATA #REQUIRED
32     xmi.tcId CDATA #IMPLIED
33 >
34 <!ELEMENT XMI.CorbaTcStruct (XMI.CorbaTcField)* >
35 <!ATTLIST XMI.CorbaTcStruct
36     xmi.tcName CDATA #REQUIRED
37     xmi.tcId CDATA #IMPLIED
38 >
39 <!ELEMENT XMI.CorbaTcField (XMI.CorbaTypeCode) >

```





- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21

```

1  <!ELEMENT XMI.CorbaTcPrincipal EMPTY >
2  <!ELEMENT XMI.CorbaTcNull EMPTY >
3  <!ELEMENT XMI.CorbaTcVoid EMPTY >
4  <!ELEMENT XMI.CorbaTcLongLong EMPTY >
5  <!ELEMENT XMI.CorbaTcLongDouble EMPTY >
6  <!-- _____ -->
7  <!-- METAMODEL: macro _____ -->
8  <!-- _____ -->
9  <!-- METAMODEL CLASS: ApplicationControlMacro _____ -->
10
11 <!ELEMENT ApplicationControlMacro.ResidencyOption EMPTY >
12 <!-- ATTLIST ApplicationControlMacro.ResidencyOption
13     xmi.value ( Resident | DynamicOption ) #REQUIRED
14 >
15 <!ELEMENT ApplicationControlMacro.isFPath EMPTY >
16 <!-- ATTLIST ApplicationControlMacro.isFPath
17     xmi.value ( False | True ) #REQUIRED
18 >
19 <!ELEMENT ApplicationControlMacro.FPathBufferSize (#PCDATA |
20     XMI.reference)* >
21 <!-- ATTLIST ApplicationControlMacro.GeneratedPSBName (#PCDATA |
22     XMI.reference)* >
23 <!-- ATTLIST ApplicationControlMacro.GeneratedPSBLanguage EMPTY >
24 <!-- ATTLIST ApplicationControlMacro.GeneratedPSBLanguage
25     xmi.value ( Assembler | COBOL | PL1 | PLI | Pascal ) #REQUIRED
26 >
27 <!-- ATTLIST ApplicationControlMacro.ProgramType EMPTY >
28 <!-- ATTLIST ApplicationControlMacro.ProgramType
29     xmi.value ( TeleProcessing | Batch ) #REQUIRED

```

```

1  >
2  <!ELEMENT ApplicationControlMacro.ProgramClass (#PCDATA |
3      XMI.reference)* >
4  <!ELEMENT ApplicationControlMacro.PSBName (#PCDATA | XMI.reference)* >
5  <!ELEMENT ApplicationControlMacro.ScheduleType EMPTY >
6  <!-- ATTLIST ApplicationControlMacro.ScheduleType
7      xmi.value ( Serial | Parallel ) #REQUIRED
8  >
9  <!ELEMENT ApplicationControlMacro.SysIDRemote (#PCDATA |
10     XMI.reference)* >
11 <!-- ATTLIST ApplicationControlMacro.SysIDLocal (#PCDATA |
12     XMI.reference)* >
13 <!-- ATTLIST ApplicationControlMacro.TRANSACT (TransactionControlMacro)* >
14 <!-- ATTLIST ApplicationControlMacro (ApplicationControlMacro.ResidencyOption?,
15     ApplicationControlMacro.isFPath?,
16     ApplicationControlMacro.FPathBufferSize?,
17     ApplicationControlMacro.GeneratedPSBName?,
18     ApplicationControlMacro.GeneratedPSBLanguage?,
19     ApplicationControlMacro.ProgramType?,
20     ApplicationControlMacro.ProgramClass?,
21     ApplicationControlMacro.PSBName?,
22     ApplicationControlMacro.ScheduleType?,
23     ApplicationControlMacro.SysIDRemote?,
24     ApplicationControlMacro.SysIDLocal?,
25     XMI.extension*,
26     ApplicationControlMacro.TRANSACT*)? >
27 <!-- ATTLIST ApplicationControlMacro
28     %XMI.element.att;
29     %XMI.link.att;
30 >
31 <!-- _____ -->
32 <!-- _____ -->
33 <!-- METAMODEL CLASS: TransactionControlMacro -->
34 <!-- _____ -->
35
36 <!-- ATTLIST TransactionControlMacro.TransactionCode (#PCDATA |
37     XMI.reference)* >

```

```

1 <!ELEMENT TransactionControlMacro.isDCLogWriteAhead EMPTY >
2 <!-- ATTLIST TransactionControlMacro.isDCLogWriteAhead
   xmi.value ( False | True ) #REQUIRED
3 >
4 <!ELEMENT TransactionControlMacro.toUpperCase EMPTY >
5 <!-- ATTLIST TransactionControlMacro.toUpperCase
   xmi.value ( False | True ) #REQUIRED
6 >
7 <!ELEMENT TransactionControlMacro.EditRoutine (#PCDATA |
8 XML.reference)* >
9
10 <!ELEMENT TransactionControlMacro.isFPath EMPTY >
11 <!-- ATTLIST TransactionControlMacro.isFPath
12 xmi.value ( False | True ) #REQUIRED
13 >
14 <!ELEMENT TransactionControlMacro.FPathBufferSize (#PCDATA |
15 XML.reference)* >
16
17 <!ELEMENT TransactionControlMacro.isInquiry EMPTY >
18 <!-- ATTLIST TransactionControlMacro.isInquiry
19 xmi.value ( False | True ) #REQUIRED
20 >
21 <!ELEMENT TransactionControlMacro.isRecover EMPTY >
22 <!-- ATTLIST TransactionControlMacro.isRecover
23 xmi.value ( False | True ) #REQUIRED
24 >
25 <!ELEMENT TransactionControlMacro.MaxRegions (#PCDATA |
26 XML.reference)* >
27
28 <!ELEMENT TransactionControlMacro.Mode EMPTY >
29 <!-- ATTLIST TransactionControlMacro.Mode
30 xmi.value ( MultipleMessage | SingleMessage ) #REQUIRED
31 >
32 <!ELEMENT TransactionControlMacro.MessageType EMPTY >
33 <!-- ATTLIST TransactionControlMacro.MessageType
34 xmi.value ( MultipleSegment | SingleSegment ) #REQUIRED
35 >
36 <!ELEMENT TransactionControlMacro.MessageIsResponse EMPTY >

```

```
1 <!ATTLIST TransactionControlMacro.MessageIsResponse
   xmi.value ( Response | NonResponse ) #REQUIRED
2 >
3 <!ELEMENT TransactionControlMacro.MessageClass (#PCDATA |
   XML.reference)* >
4 <!ELEMENT TransactionControlMacro.ParallelLimit (#PCDATA |
   XML.reference)* >
5 <!ELEMENT TransactionControlMacro.ProcessLimitCount (#PCDATA |
   XML.reference)* >
6 <!ELEMENT TransactionControlMacro.ProcessLimitSeconds (#PCDATA |
   XML.reference)* >
7 <!ELEMENT TransactionControlMacro.PriorityNormal (#PCDATA |
   XML.reference)* >
8 <!ELEMENT TransactionControlMacro.PriorityLimit (#PCDATA |
   XML.reference)* >
9 <!ELEMENT TransactionControlMacro.PriorityLimitCount (#PCDATA |
10 XML.reference)* >
11 <!ELEMENT TransactionControlMacro.PriorityLimitCount (#PCDATA |
   XML.reference)* >
12 <!ELEMENT TransactionControlMacro.isRouting EMPTY >
13 <!ATTLIST TransactionControlMacro.isRouting
   xmi.value ( False | True ) #REQUIRED
14 >
15 <!ELEMENT TransactionControlMacro.ScheduleOption (#PCDATA |
   XML.reference)* >
16 <!ELEMENT TransactionControlMacro.SegmentNumber (#PCDATA |
   XML.reference)* >
17 <!ELEMENT TransactionControlMacro.SegmentSize (#PCDATA |
   XML.reference)* >
18 <!ELEMENT TransactionControlMacro.isSerial EMPTY >
19 <!ATTLIST TransactionControlMacro.isSerial
   xmi.value ( False | True ) #REQUIRED
20 >
21 <!ELEMENT TransactionControlMacro.SPASize (#PCDATA | XML.reference)* >
```

1 <!ELEMENT TransactionControlMacro.SPAsTruncated EMPTY >  
2 <!ATTLIST TransactionControlMacro.SPAsTruncated  
xmi.value ( False | True ) #REQUIRED  
3 >  
4 <!ELEMENT TransactionControlMacro.SysIDRemote (#PCDATA |  
XMI.reference)\* >  
5 <!ELEMENT TransactionControlMacro.SysIDLocal (#PCDATA |  
XMI.reference)\* >  
6 <!ELEMENT TransactionControlMacro.isWaitForInput EMPTY >  
7 <!ATTLIST TransactionControlMacro.isWaitForInput  
xmi.value ( False | True ) #REQUIRED  
8 >  
9 <!ELEMENT TransactionControlMacro.APPLCTN (ApplicationControlMacro)? >  
10 <!ELEMENT TransactionControlMacro (TransactionControlMacro.TransactionCode?,  
TransactionControlMacro.isDCLogWriteAhead?,  
TransactionControlMacro.toUpperCase?,  
TransactionControlMacro.EditRoutine?,  
TransactionControlMacro.isFPath?,  
TransactionControlMacro.FPathBufferSize?,  
TransactionControlMacro.isInquiry?,  
TransactionControlMacro.isRecover?,  
TransactionControlMacro.MaxRegions?,  
TransactionControlMacro.Mode?,  
TransactionControlMacro.MessageType?,  
TransactionControlMacro.MessageIsResponse?,  
TransactionControlMacro.MessageClass?,  
TransactionControlMacro.ParallelLimit?,  
TransactionControlMacro.ProcessLimitCount?,  
TransactionControlMacro.ProcessLimitSeconds?,  
TransactionControlMacro.PriorityNormal?,  
TransactionControlMacro.PriorityLimit?,  
TransactionControlMacro.PriorityLimitCount?,  
TransactionControlMacro.isRouting?,  
TransactionControlMacro.ScheduleOption?,  
TransactionControlMacro.SegmentNumber?,  
TransactionControlMacro.SegmentSize?,  
TransactionControlMacro.isSerial?,  
TransactionControlMacro.SPASize?,  
TransactionControlMacro.SPAsTruncated?,  
TransactionControlMacro.SysIDRemote?,  
21

1 TransactionControlMacro.SysIDLocal?,  
TransactionControlMacro.isWaitForInput?,  
2 XMI.extension\*,  
TransactionControlMacro.APPLCTN?)? >  
3 <!ATTLIST TransactionControlMacro  
%XMI.element.att;  
%XMI.link.att;  
4 >  
5 <!ELEMENT macro ((ApplicationControlMacro | TransactionControlMacro)\*) >  
6 <!ATTLIST macro  
%XMI.element.att;  
%XMI.link.att;  
7 >  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21



Claims

1. A method for communicating with an Information Management System (IMS) using eXtensible Markup Language (XML) documents, the method comprising:

receiving a document comprising an IMS transaction definition encoded in

XML;

obtaining a Document Type Definition (DTD) specifying rules for decoding

the IMS transaction definition;

parsing the XML document using the DTD to decode the IMS transaction

definition; and

providing the decoded IMS transaction definition to the IMS.

2. The method of claim 1, wherein the IMS transaction definition comprises an APPLCTN macro.

3. The method of claim 1, wherein the IMS transaction definition comprises a TRANSACT macro.

4. The method of claim 1, wherein the DTD comprises an XML Metadata Interchange (XMI) DTD.

8. A system for communicating with an Information Management System (IMS) using eXtensible Markup Language (XML) documents, the system comprising:
- a document reception module configured to receive a document comprising
- an IMS transaction definition encoded in XML;

1 a parser configured to obtain a Document Type Definition (DTD) specifying  
2 rules for decoding the IMS transaction definition and further  
3 configured to parse the XML document using the DTD to decode the  
4 IMS transaction definition; and  
5 an IMS interface module configured to provide the decoded IMS transaction  
6 definition to the IMS.

7  
8 9. The system of claim 8, wherein the IMS transaction definition  
9 comprises an APPLCTN macro.

10  
11 10. The system of claim 8, wherein the IMS transaction definition  
12 comprises a TRANSACT macro.

13  
14 11. The system of claim 8, wherein the DTD comprises an XML Metadata  
15 Interchange (XMI) DTD.

16  
17 12. The system of claim 8, wherein the document reception module  
18 comprises an IMS gateway.

19  
20 13. The system of claim 8, further comprising:  
21

17 receiving a document comprising an IMS transaction definition encoded in  
18 XML;  
19 obtaining a Document Type Definition (DTD) specifying rules for decoding  
20 the IMS transaction definition;

1 parsing the XML document using the DTD to decode the IMS transaction

2 definition; and

3 providing the decoded IMS transaction definition to the IMS.

4

5

6 16. The article of manufacture of claim 15, wherein the IMS transaction  
7 definition comprises an APPLCTN macro.

8

9

10 17. The article of manufacture of claim 15, wherein the IMS transaction  
11 definition comprises a TRANSACT macro.

12

13

14 18. The article of manufacture of claim 15, wherein the DTD comprises an  
15 XML Metadata Interchange (XMI) DTD.

16

17

18 19. The article of manufacture of claim 15, wherein the receiving step  
19 comprises receiving the document at an IMS gateway.

20

21 20. The article of manufacture of claim 15, wherein the obtaining step  
comprises:

22

1 modeling the IMS transaction definition in a Universal Modeling Language  
2 (UML) to produce a UML object model;  
3 processing the UML object model using an XML Metadata Interchange  
4 (XMI)utility to create the DTD.  
5

6  
7 21. The article of manufacture of claim 15, the method further comprising:  
8 obtaining an IMS transaction definition;  
9 obtaining a Document Type Definition (DTD) specifying rules for encoding  
10 the IMS transaction definition; and  
11 parsing the IMS transaction definition with the DTD to encode the IMS  
12 transaction definition in an XML document.  
13  
14  
15  
16  
17  
18  
19  
20  
21

# 1 REPRESENTING IMS TRANSACTION DEFINITIONS AS XML 2 DOCUMENTS

## 3 ABSTRACT OF THE DISCLOSURE

4 A method for communicating with an Information Management System  
5 (IMS) using eXtensible Markup Language (XML) documents includes the steps of  
6 receiving a document comprising an IMS transaction definition encoded in XML;  
7 obtaining a Document Type Definition (DTD) specifying rules for decoding the IMS  
8 transaction definition; parsing the XML document using the DTD to decode the IMS  
9 transaction definition; and providing the decoded IMS transaction definition to the  
10 IMS.

11 A system for communicating with an IMS using XML documents includes  
12 a document reception module configured to receive a document comprising an IMS  
13 transaction definition encoded in XML; a parser configured to obtain a Document  
14 Type Definition (DTD) specifying rules for decoding the IMS transaction definition  
15 and further configured to parse the XML document using the DTD to decode the  
16 IMS transaction definition; and an IMS interface module configured to provide the  
17 decoded IMS transaction definition to the IMS.  
18  
19  
20  
21

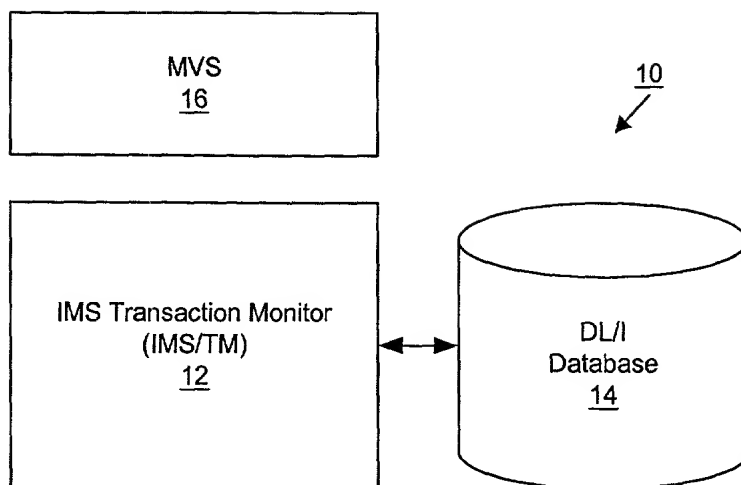


Fig. 1  
(prior art)

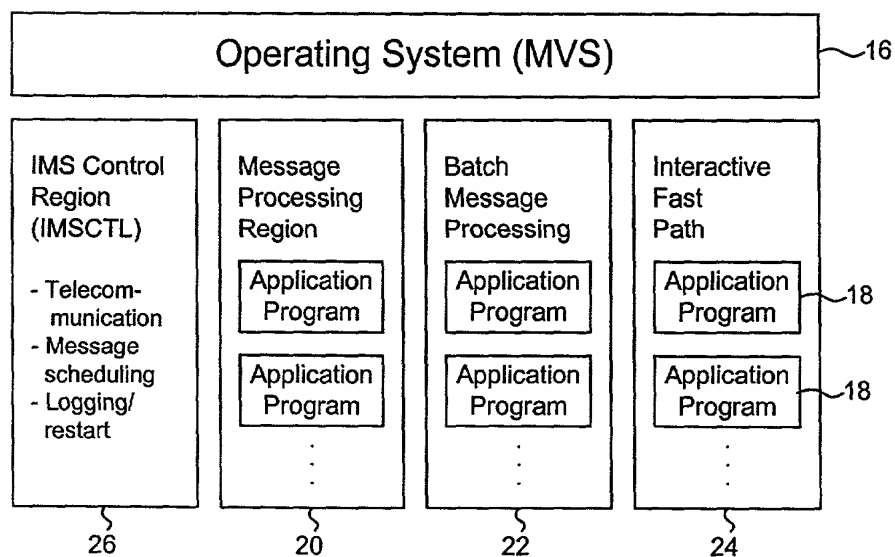


Fig. 2  
(prior art)



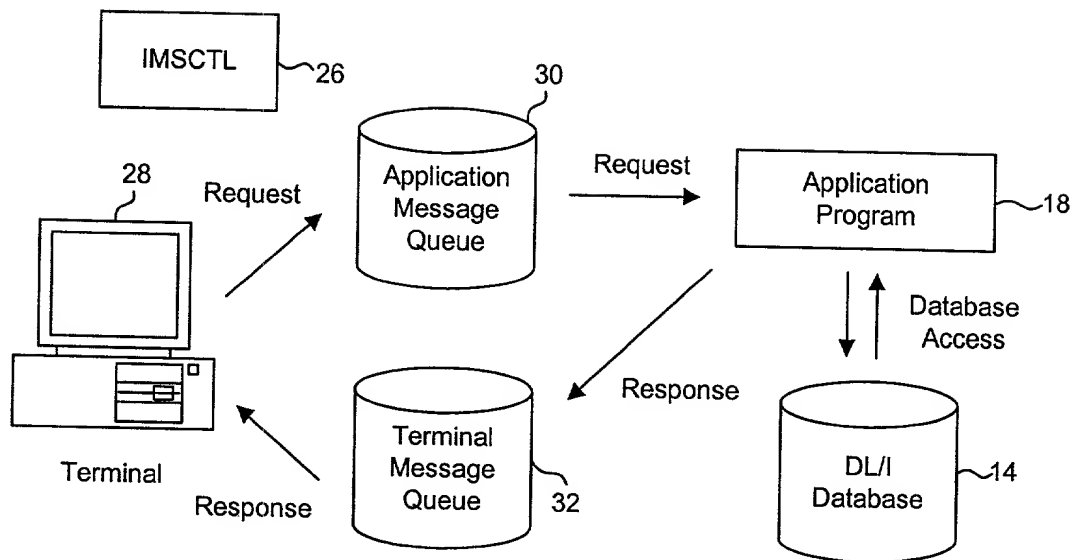


Fig. 3  
(prior art)

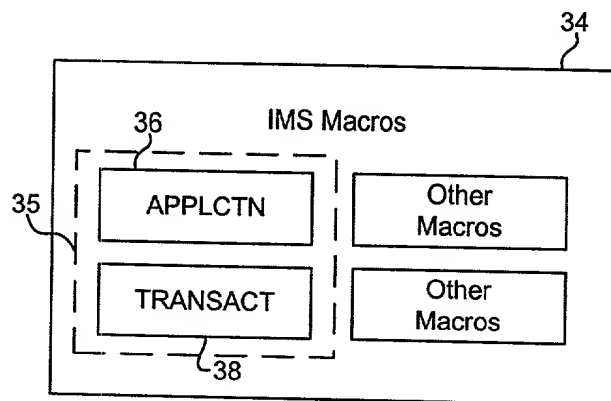


Fig. 4  
(prior art)

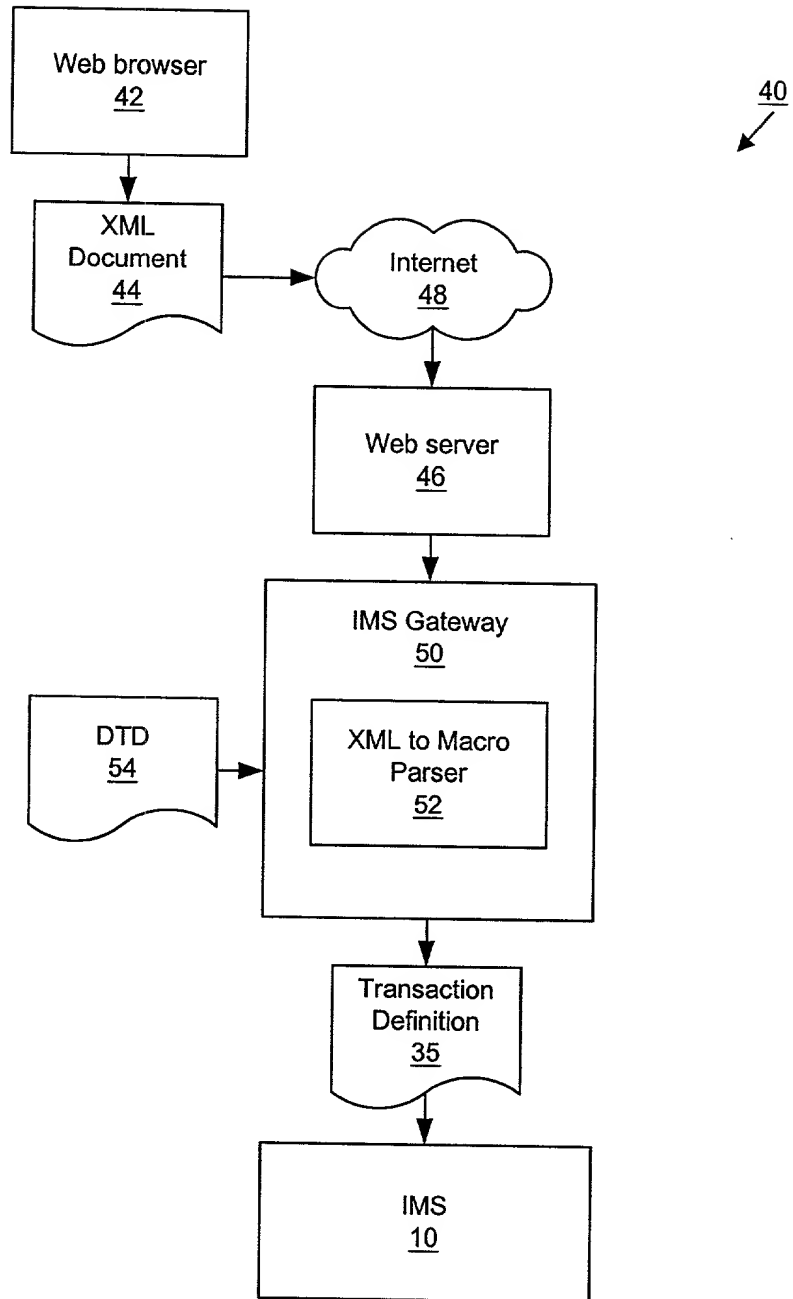


Fig. 5

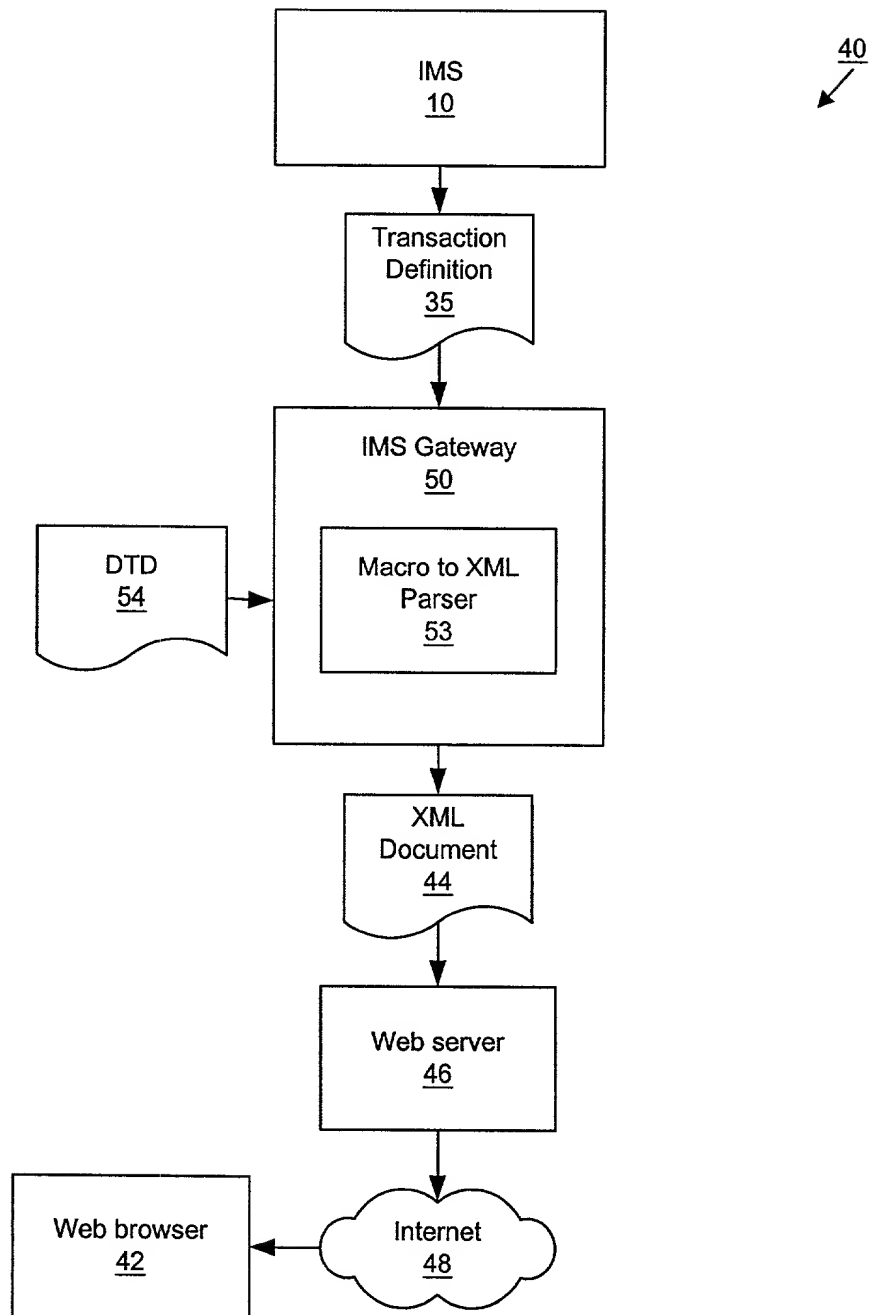


Fig. 6

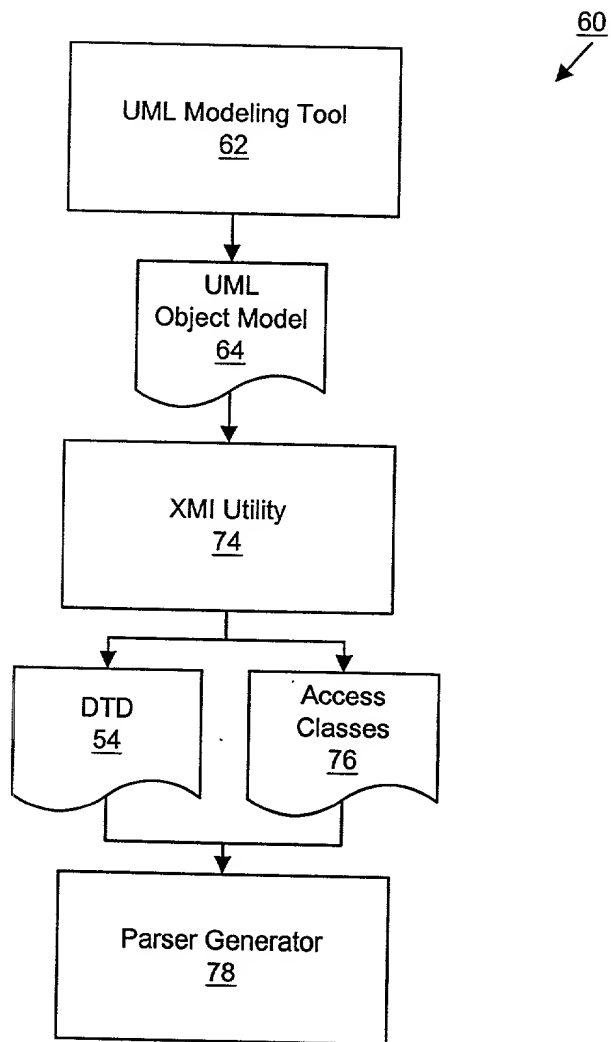


Fig. 7

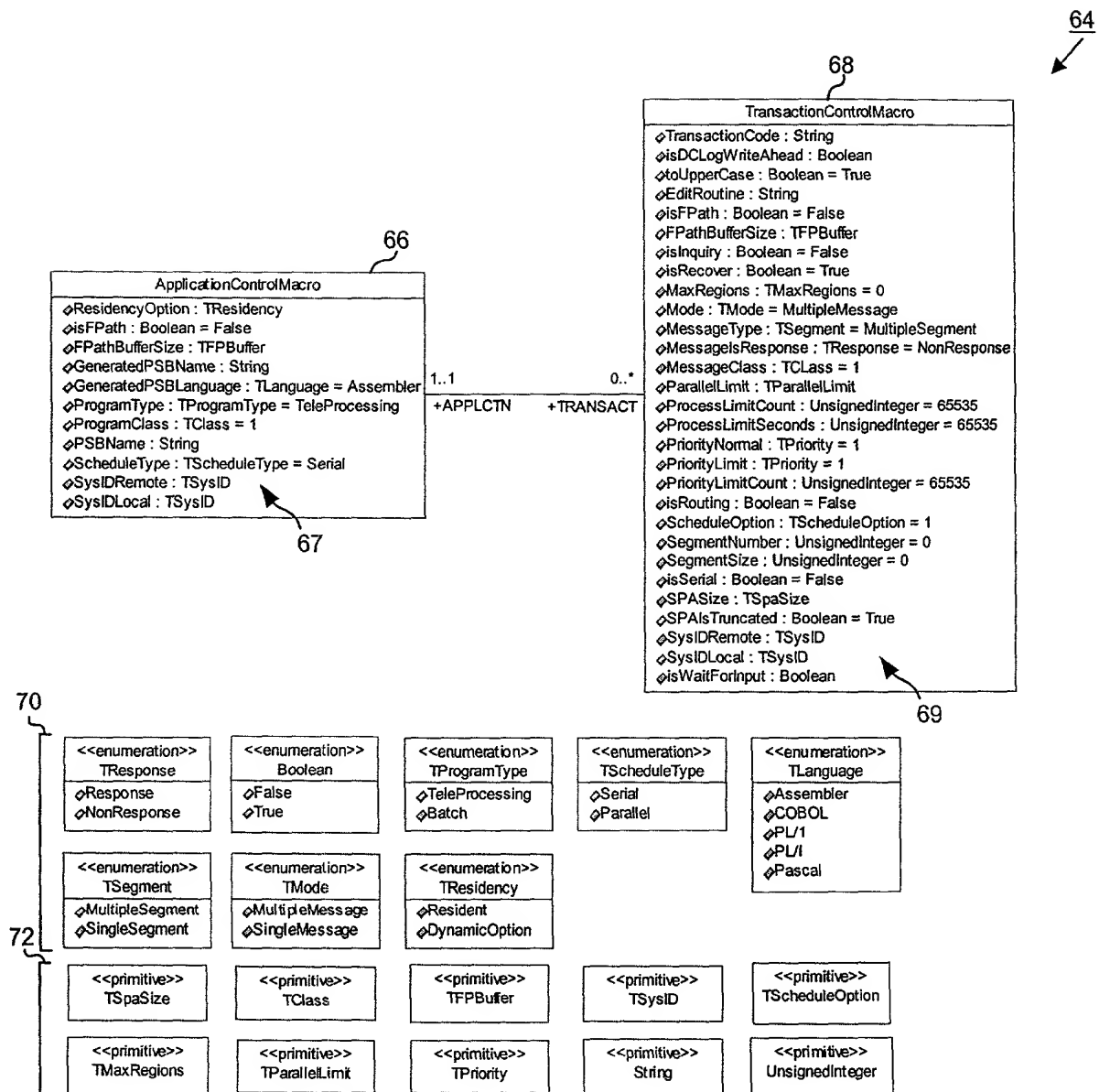


Fig. 8

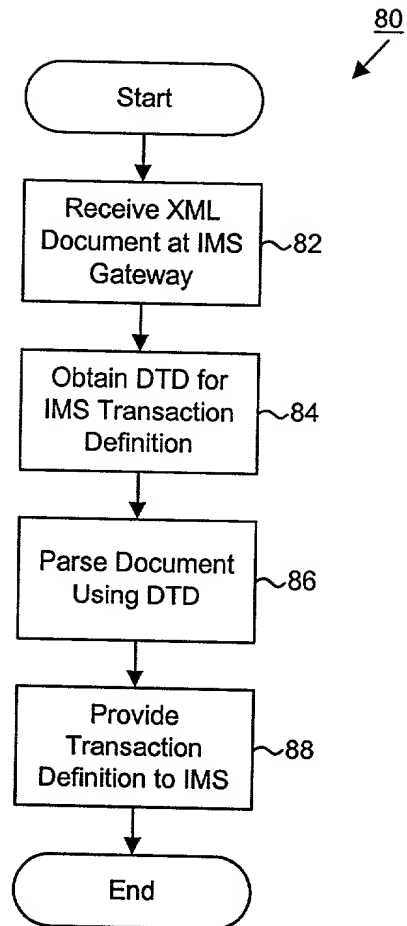


Fig. 9

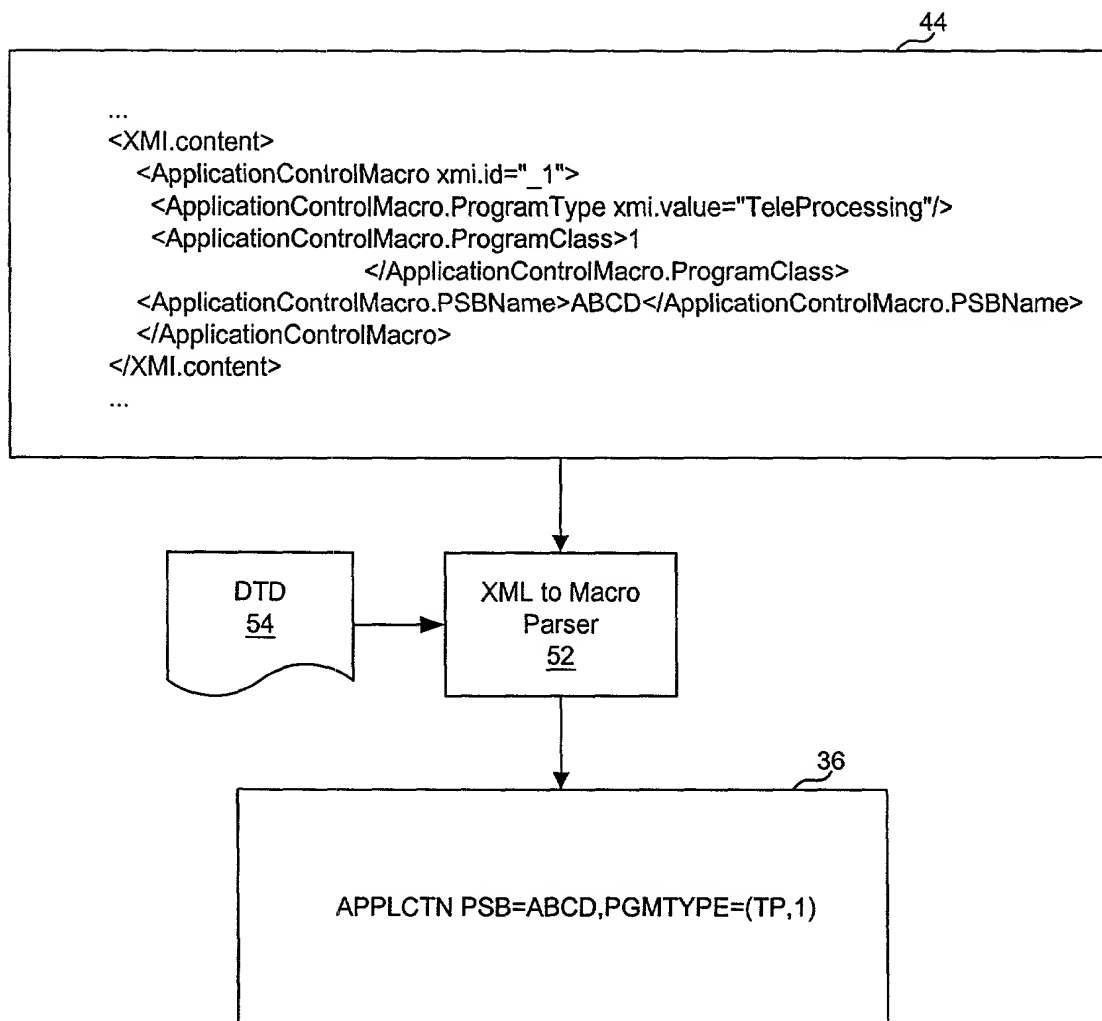


Fig. 10

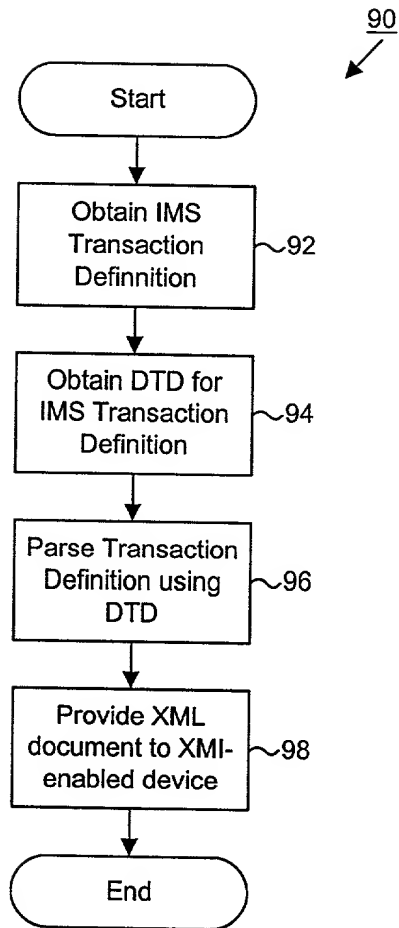


Fig. 11



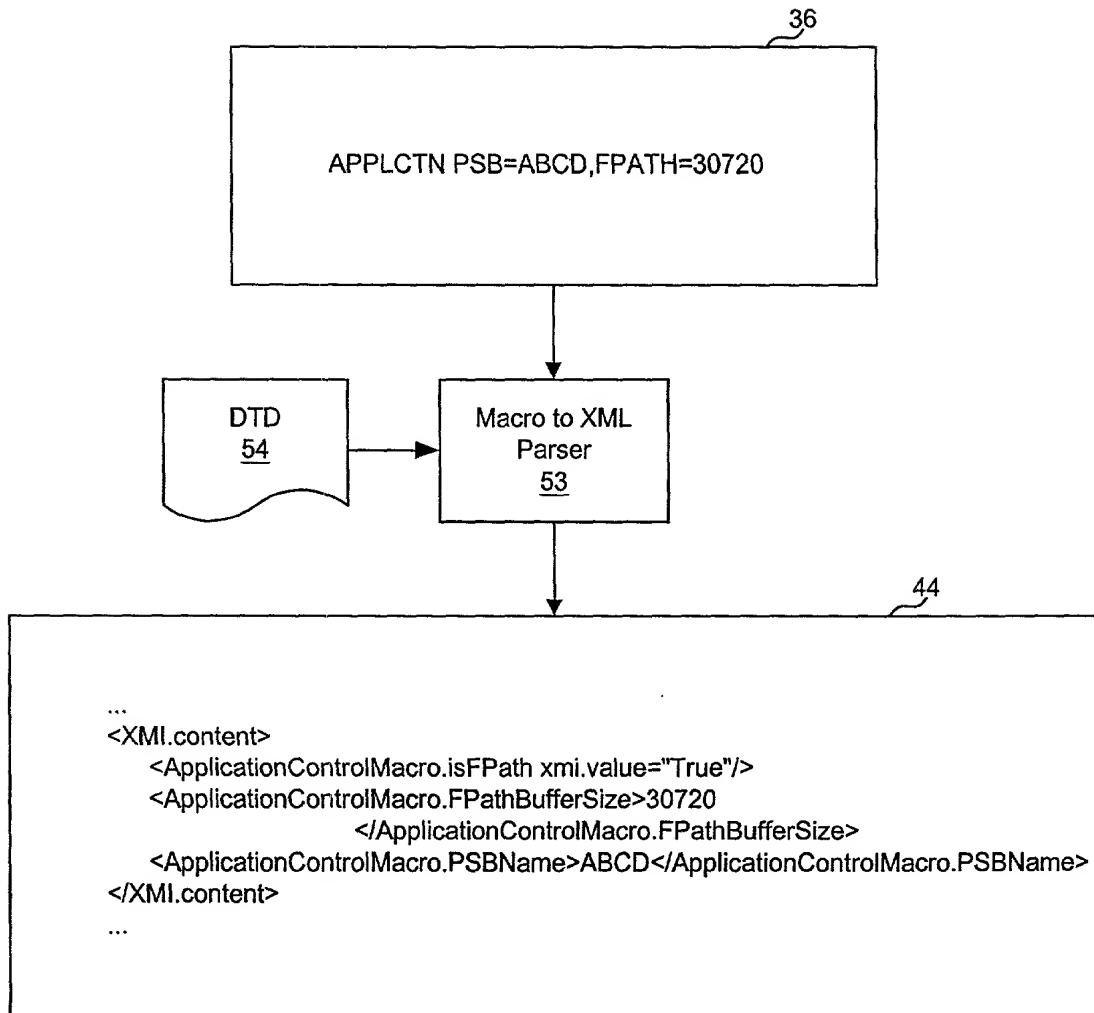


Fig. 12

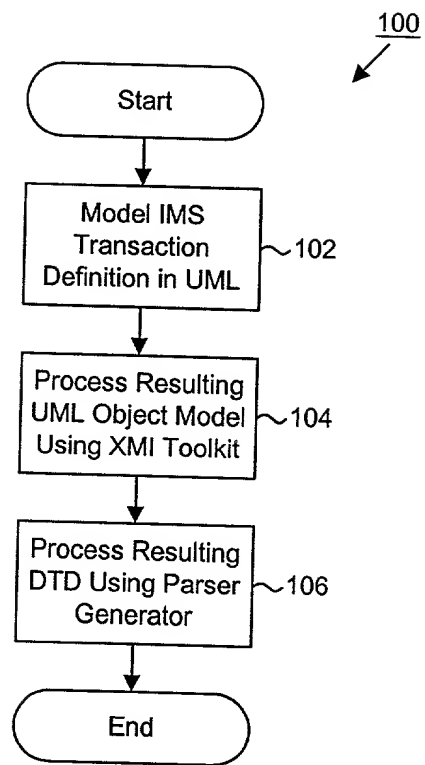


Fig. 13

**DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION**

As a below named inventor, I hereby declare that:

My residence and citizenship are as stated below next to my name;

I believe I am the original, first and joint inventor of the subject matter which is claimed and for which a patent is sought on the invention entitled

**REPRESENTING IMS TRANSACTION DEFINITIONS AS XML DOCUMENTS**

the specification of which (check one)

  X   is attached hereto.  
 \_\_\_\_\_ was filed on \_\_\_\_\_  
 as Application Serial No. \_\_\_\_\_  
 and was amended on \_\_\_\_\_ (if applicable).

I hereby stat that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to patentability as defined in Title 37, code of Federal Regulations, Section 1.56.

I hereby claim foreign priority benefits under Title 35, united States Code, Section 119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

| Prior Foreign Application(s) |           |                        | Priority Claimed                   |
|------------------------------|-----------|------------------------|------------------------------------|
| <u>  none  </u>              | _____     | _____                  | <u>      </u> Yes <u>      </u> No |
| (Number)                     | (Country) | (Day/Month/Year filed) |                                    |

I hereby claim the benefit under Title 35, Untied States Code, Section 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, Section 112, I acknowledge the duty to disclose information which is material to patentability as defined in Title 37, Code of Federal Regulations, Section 1.56, which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

|                          |                        |   |
|--------------------------|------------------------|---|
| <u>60/151,482</u>        | <u>August 30, 1999</u> | <u>Pending</u>                          |
| (Application Serial No.) | (Filing Date)          | (Status) (patented, pending, abandoned) |

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

**POWER OF ATTORNEY:** As named inventor, I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent and Trademark Office connected therewith. (list name and registration number)

Romualdas Strimaitis: 35,697  
Timothy M. Farrell: 37,321  
Ingrid M. Foerster: 36,511  
Prentiss W. Johnson: 33,123  
Christopher A. Hughes: 26,914  
John E. Hoel: 26,279  
Edward A. Pennington: 32,588  
Joseph C. Redmond, Jr.: 18,753  
Craig J. Madson: 29,407  
L. Craig Metcalf: 31,398

Evan R. Witt: 32,512  
A. John Pate: 36,234  
Gary D.E. Pierce: 38,019  
David B. Fonda: 39,672  
John R. Thompson: 40,842  
Barton W. Giddings: 41,036  
Hal D. Baird: 42,284  
Kory D. Christensen: 43,548

Send correspondence to:

Kory D. Christensen  
MADSON & METCALF  
15 West South Temple, Suite 900  
Salt Lake City, Utah 84101  
Telephone: (801) 537-1700

Full name of sole or first joint-inventor: Shyh-Mei Ho

Inventor's signature: *Shyh-Mei F. Ho* Date: *4/11/2000*

Residence: 10375 Moretti Drive, Cupertino, California 95014

Citizenship: United States of America

Post Office Address: Same

Full name of second joint-inventor: Holger Juschke

Inventor's signature: Date:

Residence: Tuttlinger Str. 110 Stuttgart, Federal Republic of Germany D-70619

Citizenship: German

Post Office Address: Same

**DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION**

As a below named inventor, I hereby declare that:

My residence and citizenship are as stated below next to my name;

I believe I am the original , first and joint inventor of the subject matter which is claimed and for which a patent is sought on the invention entitled

**REPRESENTING IMS TRANSACTION DEFINITIONS AS XML DOCUMENTS**

the specification of which (check one)

  X   is attached hereto.  
       was filed on \_\_\_\_\_  
as Application Serial No. \_\_\_\_\_  
and was amended on \_\_\_\_\_ (if applicable).

I hereby stat that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to patentability as defined in Title 37, code of Federal Regulations, Section 1.56.

I hereby claim foreign priority benefits under Title 35, united States Code, Section 119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

| Prior Foreign Application(s) |                             |                             | Priority Claimed                   |
|------------------------------|-----------------------------|-----------------------------|------------------------------------|
| <u>  none  </u>              | <u>                    </u> | <u>                    </u> | <u>      </u> Yes <u>      </u> No |
| (Number)                     | (Country)                   | (Day/Month/Year filed)      |                                    |

I hereby claim the benefit under Title 35, Untied States Code, Section 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, Section 112, I acknowledge the duty to disclose information which is material to patentability as defined in Title 37, Code of Federal Regulations, Section 1.56, which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

|                          |                        |   |
|--------------------------|------------------------|---|
| <u>60/151,482</u>        | <u>August 30, 1999</u> | <u>Pending</u>                          |
| (Application Serial No.) | (Filing Date)          | (Status) (patented, pending, abandoned) |

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

**POWER OF ATTORNEY:** As named inventor, I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent and Trademark Office connected therewith. (list name and registration number)

Romualdas Strimaitis: 35,697  
Timothy M. Farrell: 37,321  
Ingrid M. Foerster: 36,511  
Prentiss W. Johnson: 33,123  
Christopher A. Hughes: 26,914  
John E. Hoel: 26,279  
Edward A. Pennington: 32,588  
Joseph C. Redmond, Jr.: 18,753  
Craig J. Madson: 29,407  
L. Craig Metcalf: 31,398

Evan R. Witt: 32,512  
A. John Pate: 36,234  
Gary D.E. Pierce: 38,019  
David B. Fonda: 39,672  
John R. Thompson: 40,842  
Barton W. Giddings: 41,036  
Hal D. Baird: 42,284  
Kory D. Christensen: 43,548

Send correspondence to:

Kory D. Christensen  
MADSON & METCALF  
15 West South Temple, Suite 900  
Salt Lake City, Utah 84101  
Telephone: (801) 537-1700

Full name of sole or first joint-inventor: Shyh-Mei Ho

Inventor's signature:

Date:

Residence: 10375 Moretti Drive, Cupertino, California 95014

Citizenship: United States of America

Post Office Address: Same

Full name of second joint-inventor: Holger Juschke

Inventor's signature:

Date:

Residence: Tuttlinger Str. 110 Stuttgart, Federal Republic of Germany D-70619

Citizenship: German

Post Office Address: Same